AUTOMATION OF DISTRIBUTED NUMERICAL PROGRAMS CONSTRUCTION IN LUNA SYSTEM ON A MODEL APPLICATION EXAMPLE

D. Zh. Akhmed-Zaki[1,2], D. V. Lebedev[1,2], V. E. Malyshkin[3,4,5], V. A. Perepelkin[3,4]

[1]Al-Farabi Kazakh National University,
050040, Almaty, Kazakhstan
[2]University of International Business,
050010, Almaty, Kazakhstan
[3]Institute of Computational Mathematics and Mathematical Geophysics SB RAS,
630090, Novosibirsk, Russia
[4]Novosibirsk State University,
630090, Novosibirsk, Russia
[5]Novosibirsk State Technical University,
630092, Novosibirsk, Russia

The paper concerns the problem of efficient execution of a distributed numerical program, defined in a high-level language LuNA. LuNA is a programming language and a parallel program construction automation system for implementation of numerical algorithms on multicomputers (parallel computers with distributed memory). In LuNA system an application algorithm is described in a portable way, which brings the problem of its efficient execution on given hardware and data. To overcome this generally hard problem supplementary information called recommendations is demanded from the programmer. The programmer describes key properties of the application algorithm, as well as his insight on the preferred way of its efficient execution with a domain-specific language (part of the LuNA language), having no need to do low-level parallel programming, such as programming communications, resources distribution, workload balancing, etc. LuNA system takes advantage of the provided information by constructing more efficient parallel program, which implements the application algorithm. Implementation of this approach in LuNA programming system is concerned. The application algorithm is represented as an enumeration of two sets – the set of immutable pieces of data called data fragments and the set of side-effect-free sequential processes on these data called computational fragments. Computational fragments have a number of input and output data fragments assigned to define informational dependencies between computational fragments. Supplementary information (the recommendations) is provided as code annotations to express two kinds of programmer's knowledge. The first is informational recommendations, which describe key properties of the application algorithm, which are hard to obtain automatically, but are significant from the execution performance viewpoint. The second is prescriptive recommendations, which prescribe the way the application algorithm should be executed. In particular, resources distribution and computations scheduling are described. Recommendations are orthogonal to application algorithm specification in sense that they do not affect the values computed, but only influence on how the computations are conducted. In LuNA system the application algorithm is described with LuNA language (LuNA stands for Language for Numerical Algorithms), while the bodies of computational fragments are defined with sequential procedures in C++. The procedures are compiled by a conventional serial compiler, therefore LuNA compiler only has to deal with distributed execution issues. Instead of run-time interpretation of the algorithm, i. e. distributed management of data and

computational fragments, which was the case for the previous LuNA releases, the multi-agent approach is employed. With this approach each computational fragment is considered an agent, operating in a passive run-time environment. Each agent is controlled by a program, which is statically generated by LuNA-compiler (each kind of computational fragment has a different program generated). This program generally consists of migrating the agent to a remote computing node, requesting and awaiting input data fragments, compution output data fragments and doing some finalization jobs. The agent's program is formulated in C++, which allows to employ a well-developed conventional compiler to optimize it and significantly reduce run-time overhead, previously imposed by interpretation of the same logic. Despite the fact, that agents' programs are generated statically, the behavior of the multi-agent system is dynamic, leaving the room for dynamic load balancing, computations scheduling, dynamic network routing, etc. In particular, the exact node for an agent to migrate to may be defined in run-time. The performance tests were conducted on a model 3D heat equation iterative solver on a rectangular mesh. Each iteration mainly consists of solving tridiagonal equations in each of directions using the pipelined Thomas algorithm. The tests conducted showed significant performance improvement of LuNA system over its previous releases, but the reference hand-coded MPI-based implementation still outperforms LuNA in about 10 times. System code optimization and intelligent system algorithms are to be developed. However, 10 times slowdown may be tolerated in some cases due to the fact that LuNA-program development is easier and less error-prone, and that future LuNA releases will improve the performance without the need to change existing applications.

Key words: distributed programs construction automation, fragmented programming technology, LuNA system.

References

1. ANSYS Fluent Web Page / [Electron. Res.]: `https://www.ansys.com/products/fluids/ansys-fluent`, accessed: 2019.08.01.

2. Phillips, J., Braun, R., Wang, W., Gumbart, J., Tajkhorshid, E., Villa, E., Chipot, C., Skeel, R., Kale, L. Schulten, K. Scalable molecular dynamics with NAMD // Journal of Computational Chemistry. 2005. N 26. P. 1781–1802.

3. MathWorks MATLAB official web-site / [Electron. Res.]: `https://www.mathworks.com/products/matlab.html`, accessed: 2019.08.01.

4. GNU Octave Web Site / [Electron. Res.]: `https://www.gnu.org/software/octave/`, accessed: 2019.08.01.

5. WOLFRAM MATHEMATICA Web Site / [Electron. Res.]: `http://www.wolfram.com/mathematica/`, accessed: 20195.04.01.

6. Robson, M., Buch, R., Kale, L. Runtime Coordinated Heterogeneous Tasks in Charm++ // In: Proceedings of the Second Internationsl Workshop on Extreme Scale Programming Models and Middleware, 2016.

7. Wu W., Bouteiller A., Bosilca G., Faverge M., Dongarra J. Hierarchical DAG Scheduling for Hybrid Distributed Systems // In: 29th IEEE International Parallel & Distributed Processing Symposium, 2014.

8. Bauer, M., Treichler, S., Slaughter, E., Aiken, A. Legion: Expressing Locality and Independence with Logical Regions // In: the International Conference on Supercomputing (SC 2012), 2012.

9. Malyshkin, V., Perepelkin, V. LuNA Fragmented Programming System, Main Functions and Peculiarities of Run-Time Subsystem // In: Parallel Computing Technologies. LNCS. 2011. N 6873. P. 53–61.

10. Sterling, T., Anderson, M., Brodowicz, M. A Survey: Runtime Software Systems for High Performance Computing // Supercomputing Frontiers and Innovations: an International Journal. 2017. N 4 (1). P. 48–68. DOI: 10.14529/jsfi170103.

11. Thoman, P., Dichev, K., Heller, T. et al. A taxonomy of task-based parallel programming technologies for high-performance computing // The Journal of Supercomputing. 2018. N 74 (4). P. 1422–1434. DOI: 10.1007/s11227-018-2238-4.

12. Valkovsky, V., Malyshkin, V. Synthesis of parallel programs and systems on the basis of computational models. Nauka, Novosibirak, 1988.

13. Akhmed-Zaki, D., Lebedev, D., Perepelkin, V. // J Supercomput, 2018. [Electron. Res.]: `https://doi.org/10.1007/s11227-018-2710-1`.

14. Sapronov, I., Bykov, A.: Parallel pipelined algorithm // Atom 2009. N 44. P. 24–25 (in Russian).

15. Joint Supercomputing Centre of Russian Academy of Sciences Official Site / [Electron. Res.]: `http://www.jscc.ru/`, accessed: 2019.08.01.

АВТОМАТИЗАЦИЯ КОНСТРУИРОВАНИЯ РАСПРЕДЕЛЕННЫХ ПРОГРАММ ЧИСЛЕННОГО МОДЕЛИРОВАНИЯ В СИСТЕМЕ LUNA НА ПРИМЕРЕ МОДЕЛЬНОЙ ЗАДАЧИ

Д. Ж. Ахмед-Заки[1,2], Д. В. Лебедев[1,2], В. Э. Малышкин[3,4,5], В. А. Перепелкин[3,4]

[1]Казахский национальный университет им. аль-Фараби,
050040, Алма-Ата, Казахстан
[2]Университет международного бизнеса,
050010, Алма-Ата, Казахстан
[3]Институт вычислительной математики и математической геофизики СО РАН,
630090, Новосибирск, Россия
[4]Новосибирский национальный исследовательский государственный университет,
630090, Новосибирск, Россия
[5]Новосибирский государственный технический университет,
630092, Новосибирск, Россия

УДК 004.4'2
DOI: 10.24411/2073-0667-2019-00017

В статье рассматривается проблема эффективного распределенного исполнения фрагментированных программ в системе LuNA — системе автоматизации конструирования параллельных программ численного моделирования для мультикомпьютеров. В системе LuNA прикладной алгоритм описывается на языке высокого уровня, что делает это описание переносимым, но при этом встает сложная проблема обеспечения удовлетворительной эффективности исполнения этого алгоритма на заданном оборудовании и данных. Для преодоления этой проблемы привлекается дополнительное знание пользователя о структуре и свойствах алгоритма, а также о рекомендуемых способах его эффективного параллельного исполнения. Это знание формулируется в виде аннотаций к описанию алгоритма, называемых рекомендациями. При таком подходе пользователю не приходится программировать сложную распределенную логику и управление, а система использует знания пользователя для повышения эффективности работы. Рассматривается реализация этого подхода в системе LuNA. Представлены результаты сравнительного экспериментального исследования производительности.

Ключевые слова: автоматизация конструирования параллельных программ, технология фрагментированного программирования, система LuNA.

Введение. Существенный прирост вычислительных мощностей суперкомпьютеров в последние десятилетия сопровождается усложнением использования этого высокопроизводительного вычислительного оборудования его пользователями — прикладными программистами в области научного численного моделирования. Эффективное использование

Работа поддержана проектом BR05236340 „Создание высокопроизводительных интеллектуальных технологий анализа и принятия решения для системы „логистика-агломерация“ в рамках формирования цифровой экономики РК“.

современных суперкомпьютерных ресурсов подразумевает, что прикладное программное обеспечение должно быть масштабируемым и настраиваемым на конфигурацию оборудования и данные. (Тут и далее эффективность понимается в смысле нефункциональных свойств программы — времени выполнения, расхода памяти, нагрузки на коммуникационную сеть и т. п.). В ряде случаев требуется динамическая балансировка нагрузки, поддержка сопроцессоров (GPU, FPGA и т. п.), отказоустойчивость и другие нефункциональные свойства. Поддержка таких свойств является сложной задачей и требует от программиста специфичных знаний и умений, отличных от тех, что требуются при разработке „вычислительной“ части программы.

Особенно эта проблема затрагивает пользователей, разрабатывающих новые численные модели и алгоритмы, когда нет возможности использовать существующие отлаженные и оптимизированные программные инструменты, такие как ANSYS Fluent [1], NAMD [2] и пр. Строгие ограничения по производительности и расходу памяти, как правило, не позволяют использовать математическое программное обеспечение, ориентированное на неспециалистов в области системного параллельного программирования (MathWorks MATLAB [3], GNU OCTAVE [4], Wolfram Mathematica [5] и т. п.). Единственным вариантом снижения сложности разработки эффективных параллельных программ остается использование систем параллельного программирования [6–11], автоматизирующих многие низкоуровневые рутинные задачи параллельного программирования, которые часто вызывают затруднение при ручном программировании. Эти системы предоставляют более высокоуровневые средства (чем традиционные языки программирования), пригодные в различных частных случаях.

В системе Charm++ [6] вычисления представляются в виде множества распределенных взаимодействующих посредством сообщений объектов, называемых чарами (chare). Среда исполнения способна передавать чары по сети посредством сериализации, тем самым изменяя их распределение по вычислительным узлам, планировать (переупорядочивать) обработку сообщений и выполнять другие задачи управления исполнением программы, направленные на повышение эффективности. Пользователь имеет некоторые возможности настройки процесса исполнения программы, включая выбор алгоритма динамической балансировки нагрузки. Система Charm++ позволяет достичь высокой эффективности исполнения программ, освобождая при этом пользователя от ряда сложных задач параллельного программирования. В системе PaRSEC [7] область применения ограничена классом алгоритмов линейной алгебры над плотными матрицами (и сходными алгоритмами). В частности, итерации с динамическим условием не поддерживаются. Это и другие ограничения использованы для повышения эффективности работы системных алгоритмов и эвристик в обозначенной предметной области. Система Legion [8] применяет перспективный подход к раздельному описанию вычислительной части и управления ходом вычислений как двух ортогональных частей программы. При этом подходе пользователь несет ответственность за распределение ресурсов, планирование вычислений и другие задачи управления вычислениями, но средства системы Legion позволяют делать это без риска внесения ошибок в вычислительную часть программы. Система LuNA [9] использует похожий подход, но дополнительно предоставляет возможность автоматизации конструирования управляющего кода. Многие другие системы существуют и развиваются, исследуя различные подходы, вычислительные модели, системные алгоритмы и эвристики, чтобы предоставить лучшие средства автоматизации конструирования параллельных программ [10, 11]. Несмотря на большое количество усилий, вложенных в создание и развитие таких

систем, еще многое должно быть сделано для расширения закрываемых ими предметных областей и повышения качества выполняемой автоматизации.

В статье рассматривается подход, применяемый в системе LuNA для достижения удовлетворительной производительности конструируемых программ. Система LuNA предназначена для автоматизации конструирования параллельных программ, реализующих крупномасштабные численные модели для суперкомпьютеров. Система разрабатывается в ИВМиМГ СО РАН. Рассматриваемый в работе подход является приложением технологии активных знаний к конкретной предметной области — автоматизации конструирования параллельных программ с заданными свойствами.

1. Подход технологии фрагментированного программирования. В технологии фрагментированного программирования прикладной алгоритм представляется в аппаратно-независимой форме, называемой фрагментированным алгоритмом (ФА). ФА является декларативной спецификацией, которая определяет два потенциально бесконечных множества — множество фрагментов вычислений (ФВ) и множество фрагментов данных (ФД). Каждый ФВ — это вычислительный процесс, задаваемый чистой (без „побочных эффектов“) процедурой, а ФД — это иммутабельная (единственного присваивания) переменная, значением которой может выступать некоторая часть данных прикладного алгоритма (например, целое число или домен вычислительной сетки на заданном временном шаге). Для каждого ФВ задается два непересекающихся конечных подмножества ФД, называемых множеством входных ФД и выходных ФД для данного ФВ. Процедура, задающая вычислительный процесс ФВ, вычисляет значения выходных ФД из значений входных ФД при условии, что значения входных ФД находятся в памяти узла, на котором находится ФВ. ФА является представлением, перечисляющим множества ФВ и ФД с помощью ряда операторов. Это представление основывается на определении вычислительных моделей [12], а именно, ФА является вычислительной моделью частного вида, на которой выводим единственный алгоритм.

Фрагментированная программа (ФП) — это ФА с дополнительной информацией, называемой рекомендациями. ФА определяет функциональную часть вычислений (т. е. как одни ФД вычисляются из других), в то время как рекомендации влияют на нефункциональные свойства вычислительного процесса, такие как время вычислений, расход памяти, нагрузка на сеть и т. п. Например, рекомендации могут предписывать двум ФД использовать один и тот же буфер памяти в различные промежутки времени с целью уменьшения расхода памяти; или рекомендации могут определять стратегию размещения элементов распределенного массива.

ФА и рекомендации ортогональны в том смысле, что рекомендации не влияют на вычисляемые значения, а только на то, как фрагменты (ФД и ФВ) отображаются на ресурсы мультикомпьютера во времени. Разные рекомендации могут использоваться для отображения на ФА на мультикомпьютеры различных конфигураций и/или с учетом различных оптимизационных критериев (память, время, сеть и т. п.). Рекомендации делятся на два вида — информационные и предписывающие. Информационные рекомендации описывают свойства ФА, которые сложно выявить автоматически, но которые важны с точки зрения обеспечения эффективности исполнения ФА. Примерами информационных рекомендаций могут служить оценочная сложность алгоритма или структура данных ФА. Предписывающие рекомендации направляют вычисления тем или иным способом, например, задают отображение ФД на вычислительные узлы или определяют порядок выполнения ФВ (в рамках информационных зависимостей между ними). Рекомендации обоих видов не яв-

ляются обязательными, и даже если они присутствуют, то могут быть проигнорированы системой частично или полностью.

Подобная ортогональность встречается в различных системах программирования [6–9], т. к. является основой, позволяющей системам контролировать исполнение программы. Проиллюстрируем различия на примере распределения объектов (фрагментов, задач и т.п.) по узлам мультикомпьютера. В некоторых системах, таких как Charm++ объекты распределяются системными алгоритмами. В других, таких как PaRSEC, пользователь задает распределение, но не программирует его. В системах, таких как Legion, пользователь задает распределение, программируя соответствующую логику с использованием системного программного интерфейса (API). В системе LuNA используется гибридный подход. Если рекомендации отсутствуют, то система определяет распределение имеющимися в ней алгоритмами. Если информационные рекомендации предоставлены, то (вероятно) более качественное распределение будет построено с учетом этого знания. Если имеются предписывающие рекомендации, то система будет следовать им (т. е. пользователь задает распределение, не программируя его). Предписывающие рекомендации наименее переносимы, но полезны, пока системные алгоритмы не развиты настолько, чтобы конструировать конкурентоспособное распределение автоматически. С определенного момента предписывающие рекомендации следует отбросить. Информационные рекомендации полезны на протяжении более долгого времени — до тех пор, пока соответствующая информация не может быть извлечена системой автоматически (статически, из описания ФА или динамически, на основе профилирования и трассировки исполнения).

В соответствии с этим подходом ФА не содержит решений, связанных с нефункциональными свойствами, такими как множественное присваивание (мутабельность данных), порядок вычислений (за исключением информационных зависимостей), распределение ресурсов, сборка мусора и т.п. В системе Charm++, например, множественное присваивание присутствует с целью оптимизации производительности, но в перспективе станет препятствием в достижении высокой производительности в существующих Charm++ программах. Рекомендации на текущий момент играют критическую роль в системе LuNA для достижения высокой производительности, потому что знания по автоматизации конструирования параллельных программ, заложенные в систему, недостаточны для полностью автоматического эффективного исполнения ФА. Рекомендации закрывают недостаток этих знаний, позволяя достигать эффективного исполнения ФА с помощью дополнительных усилий.

2. **Система LuNA.** В системе LuNA ФП описывается с помощью двух языков программирования — LuNA (Language for Numerical Algorithms) и C++. LuNA используется для описания множеств ФД, ФВ и рекомендаций, а C++ — для описания последовательных подпрограмм, задающих процессы ФВ. Язык C++ является мощным традиционным языком, хорошо поддержанным эффективными оптимизирующими компиляторами и прочими инструментами, что позволяет делать отдельные ФВ высокоэффективными, оставляя на систему LuNA лишь вопросы распределенного исполнения ФА.

Предыдущие версии системы LuNA использовали подход полуинтерпретации, когда ФП интерпретируется средой исполнения, а исполнение отдельных ФВ осуществляется путем запуска последовательного скомпилированного кода. Среда исполнения конструирует внутренние системные объекты, соответствующие ФД и ФВ, распределяет их по вычислительным узлам мультикомпьютера, обеспечивает передачу по сети входных ФД на нужные узлы и т. д. Текущая версия системы использует принципиально тот же, но на

практике более эффективный подход. При этом подходе каждый ФВ рассматривается как агент мультиагентной системы, управляемый статически сгенерированной программой. Разные типы ФВ управляются разными программами. Среда исполнения в этом случае является пассивным окружением, реагирующим на команды агента. Программа агента генерируется статически компилятором LuNA и обычно включает следующие основные шаги:

— миграция на другой узел (при необходимости), на котором ФВ будет исполнен,
— запрос входных ФД с других узлов и ожидание их получения,
— исполнение вычислительной процедуры над входными ФД,
— порождение новых (последующих) ФВ,
— выполнение завершающих действий.

Завершающие действия могут включать удаление ФД, сохранение выходных ФД на текущий или удаленный узлы и т. п. Определенные шаги зависят от типа ФВ (исполнение одиночного ФВ, исполнение фрагментированной подпрограммы, цикл FOR- или WHILE-типа, условный оператор и т. п.), поддерживаемого языком LuNA. Программа ФВ зависит также от алгоритмов компилятора, рекомендаций, конфигурации оборудования и т. п. Как правило, все решения о ходе исполнения ФА, которые можно принять статически, формулируются в виде программ ФВ. Отметим, что программы ФВ не являются жесткими. Например, этап миграции определяется статически, но конкретный узел и маршрут миграции могут определяться динамически. Все решения об исполнении ФА, которые должны быть приняты во время исполнения, остаются незафиксированными в программах ФВ.

В связи с тем, что программы ФВ формулируются на языке C++, они компилируются традиционным компилятором, и поэтому к ним оказывается применено множество небольших, но важных оптимизаций, таких как статическое вычисление выражений, устранение мертвого кода, оптимизации стека вызовов и т. п. В случае полуинтерпретации все эти действия должны были быть выполнены в полном объеме, что существенно уменьшало общую производительность системы, особенно при большом количестве фрагментов.

В то время как оптимизация последовательного кода (процедур, задающих вычисления ФВ и сгенерированные программы ФВ) отдается хорошо разработанным традиционным компиляторам, компилятор и среда исполнения LuNA фокусируются на вопросах распределенного исполнения. На основе рекомендаций решения по распределению ФВ и ФД по вычислительным узлам, порядок выполнения ФВ, сборка мусора и пр. принимаются статически (в LuNA компиляторе) или динамически в среде исполнения. Рассмотрение соответствующих алгоритмов выходит за рамки статьи и может быть найдено в других публикациях по системе LuNA.

3. **Тестирование производительности.** Для исследования производительности исполнения LuNA-программ был проведен ряд тестов. В качестве приложения был выбран решатель модельного трехмерного уравнения теплопроводности в единичном кубе. Это приложение было рассмотрено в статье [13]. Данные приложения составляяет трехмерная прямоугольная сетка с трехмерной декомпозицией на домены. Вычисления осуществляются итерационно, каждая итерация решается конвейерным алгоритмом прогонки по каждому из трех направлений. Структурно это приложение соответствует широкому классу итерационных приложений на регулярных сетках.

Тестирование проводилось на кластере MVS-10p МСЦ РАН [15], включающем узлы на базе двух процессоров Xeon E5-2690 с 64 ГБ ОЗУ. Следующие репрезентативные для приложений данного класса параметры были выбраны. Размер сетки — от 1003 до 10003

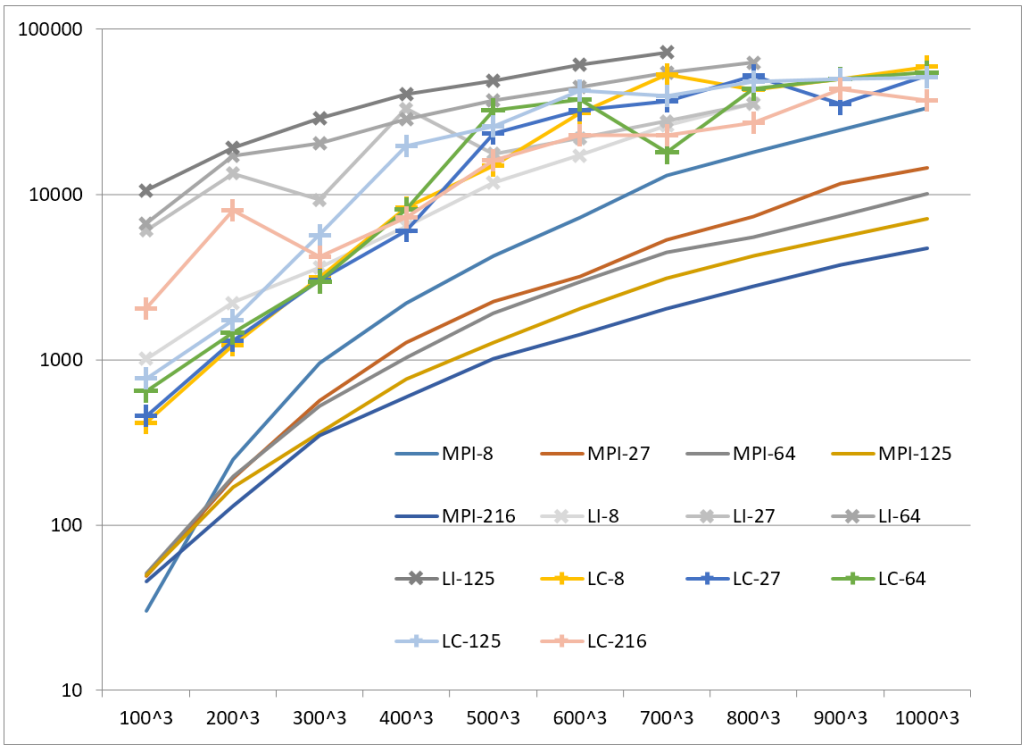

Рис. 1. Время выполнения программы (с). LI — интерпретатор ФП, LC — предкомпилированные программы ФВ, MPI — ручная реализация средствами MPI. Число означает количество ядер. По оси X задан размер сетки

с шагом 100 по каждому измерению. Количество ядер: от 23 (8) до 63 (216) с шагом 1 (в каждом измерении).

Результаты представлены на рис. 1. Линии LI (LuNA-Interpreter) соответствуют предыдущей версии системы (полуинтерпретация), линии LC (LuNA-Compiled) соответствуют текущей версии системы (мультиагентный подход с генерацией программ ФВ), а линии MPI соответствуют ручной реализации того же алгоритма на основе MPI. MPI-реализация может рассматриваться как эталонный (по производительности) вариант. Достижение в автоматически сгенерированных программах производительности, сравнимой с производительностью программ, написанных вручную, можно считать достаточным.

Из рис. 1 видно, что текущая версия системы LuNA обеспечивает существенно более производительное исполнение, чем вариант с полуинтерпретацией, хотя и проигрывает в производительности ручной реализации. Также видно, что преимущество текущей версии больше для фрагментов небольших размеров, что ожидаемо, т. к. оптимизация программ ФВ традиционным компилятором снижает, главным образом, накладные расходы, связанные с количеством фрагментов, а не их размером. Ручная реализация оказывается примерно на порядок быстрее, что говорит о необходимости дальнейшего развития системных алгоритмов. В частности, накладные расходы на сеть, вызванные системными коммуникациями среды исполнения, требуют оптимизации. Тем не менее, даже такая эффективность может являться терпимой, если принять во внимание, во-первых, снижение сложности разработки параллельной программы, и, во-вторых, тот факт, что по мере

оптимизации системы LuNA существующие ФП будут ускоряться без необходимости что-либо модифицировать в них.

Заключение. Рассмотрен подход к достижению эффективности исполнения параллельных программ, описанных на высоком уровне абстракции, а также его реализация в системе автоматизации конструирования параллельных программ LuNA. Проведено исследование производительности на примере модельной итерационной задачи на трехмерной сетке. Выполнено сравнение с ручной реализацией того же прикладного алгоритма. В будущем планируется как оптимизация системного кода, так и разработка интеллектуальных системных алгоритмов с целью достижения более высокой эффективности исполнения фрагментированных алгоритмов.

Список литературы

1. ANSYS Fluent Web Page / [Electron. Res.]: `https://www.ansys.com/products/fluids/ansys-fluent`, accessed: 2019.08.01.

2. Phillips, J., Braun, R., Wang, W., Gumbart, J., Tajkhorshid, E., Villa, E., Chipot, C., Skeel, R., Kale, L. Schulten, K. Scalable molecular dynamics with NAMD // Journal of Computational Chemistry. 2005. N 26. P. 1781–1802.

3. MathWorks MATLAB official web-site / [Electron. Res.]: `https://www.mathworks.com/products/matlab.html`, accessed: 2019.08.01.

4. GNU Octave Web Site / [Electron. Res.]: `https://www.gnu.org/software/octave/`, accessed: 2019.08.01.

5. WOLFRAM MATHEMATICA Web Site / [Electron. Res.]: `http://www.wolfram.com/mathematica/`, accessed: 2019.04.01.

6. Robson, M., Buch, R., Kale, L.: Runtime Coordinated Heterogeneous Tasks in Charm++ // In: Proceedings of the Second Internationsl Workshop on Extreme Scale Programming Models and Middleware, 2016.

7. Wu W., Bouteiller A., Bosilca G., Faverge M., Dongarra J. Hierarchical DAG Scheduling for Hybrid Distributed Systems // In: 29th IEEE International Parallel & Distributed Processing Symposium, 2014.

8. Bauer, M., Treichler, S., Slaughter, E., Aiken, A. Legion: Expressing Locality and Independence with Logical Regions // In: the International Conference on Supercomputing (SC 2012), 2012.

9. Malyshkin, V., Perepelkin, V. LuNA Fragmented Programming System, Main Functions and Peculiarities of Run-Time Subsystem // In: Parallel Computing Technologies. LNCS. 2011. N 6873. P. 53–61.

10. Sterling, T., Anderson, M., Brodowicz, M. A Survey: Runtime Software Systems for High Performance Computing // Supercomputing Frontiers and Innovations: an International Journal. 2017. N 4 (1). P. 48–68. DOI: 10.14529/jsfi170103.

11. Thoman, P., Dichev, K., Heller, T. et al. A taxonomy of task-based parallel programming technologies for high-performance computing // The Journal of Supercomputing. 2018. N 74 (4). P. 1422–1434. DOI: 10.1007/s11227-018-2238-4.

12. Valkovsky, V., Malyshkin, V. Synthesis of parallel programs and systems on the basis of computational models. Nauka, Novosibirak, 1988.

13. Akhmed-Zaki, D., Lebedev, D., Perepelkin, V. // J Supercomput, 2018. [Electron. Res.]: `https://doi.org/10.1007/s11227-018-2710-1`.

14. Сапронов И., Быков А. Параллельно-конвейерный алгоритм // Атом 2009. № 44. С. 24–25.

15. Joint Supercomputing Centre of Russian Academy of Sciences Official Site / [Electron. Res.]: http://www.jscc.ru/, accessed: 2019.08.01.

Дархан Жумаканович Ахмед-Заки — доктор технических наук, президент Университета международного бизнеса, Алма-Ата, Казахстан. E-mail: darhan_a@mail.ru, тел.: +7 (727) 259 8000. В 2002 году окончил механико-математический факультет Казахского национального университета им. аль-Фараби по специальности „Прикладная математика“.

В 2002–2006 годах обучался в магистратуре и аспирантуре механико-математического факультета Казахского национального университета имени аль-Фараби.

В 2010 году защитил докторскую диссертацию по специальности 05.13.18 „Математическое моделирование, численные методы и комплексы программ“.

Является автором более 100 научных работ, опубликованных в ведущих журналах Республики Казахстан и дальнего зарубежья. Основные области научных интересов: организация параллельных и распределенных вычислений, математическое моделирование физических процессов, теория тепловой фильтрации несжимаемых жидкостей в пористой среде и верификация программ.

Akhmed-Zaki Darkhan Zhumakanovich — doctor of technical sciences, president of University of International Business, Almaty, Kazakhstan. E-mail: darhan_a@mail.ru, tel.: +7 (727) 259 8000.

In 2002 graduated from mechanical-mathematical faculty of al-Farabi Kazakh National University in the applied mathematics specialty.

In 2002-2006 studied in magistracy and postgraduate study of al-Farabi Kazakh National University.

In 2010 defended his doctoral dissertation in the specialty 05.13.18 – mathematical simulation, numerical methods and software complexes. Is an author of more than 100 papers, published in leading journals of Republic of Kazakhstan and other countries. Main areas of scientific interests: organization of parallel and distributed computations, mathematical simulation of physical phenomena, theory of thermal filtration of incompressible fluids in porous medium, software verification.

Лебедев Данил Владимирович — PhD, заведующий лабораторией цифровых технологий Университета международного бизнеса, Алма-Ата, Казахстан. E-mail: dan-lebedev@mail.ru, тел.: +7 727 259 8000.

Окончил магистратуру Казахского национального университета им. аль-Фараби в 2004 году. В 2016 окончил докторантуру PhD в том же университете. Основные области научных интересов: параллельные и распределенные вычисления, задачи фильтрации, машинное обучение.

Lebedev Danil Vladimirovich — PhD, head of digital technologies laboratory in University of International Business, Almaty, Kazakhstan. E-mail: dan-lebedev@mail.ru, tel.: +7 (727) 259 8000.

Graduated with the master of sciences degree from al-Farabi Kazakh National University in 2004. In 2016 graduated the PhD post graduate in the same university. Main areas of scientific interest: parallel and distributed computations, filtration problems, machine learning.

Виктор Эммануилович Малышкин — получил степень магистра математики в Томском государственном университете (1970), степень доктора технических наук в Новосибирском государственном университете (1993). В настоящее время является заведующим лабораторией синтеза параллельных программ в Институте вычислительной математики и математической геофизики СО РАН. Он также основал и в настоящее время возглавляет кафедру параллельных вычислений

в Национальном исследовательском университете Новосибирска. Является одним из организаторов международной конференции PaCT (Parallel Computing Technologies), проводимых каждый нечетный год в России.

Имеет более 110 публикаций по параллельным и распределенным вычислениям, синтезу параллельных программ, суперкомпьютерному программному обеспечению и приложениям, параллельной реализации крупномасштабных численных моделей.

В настоящее время область его интересов включает параллельные вычислительные технологии, языки и системы параллельного программирования, методы и средства параллельной реализации крупномасштабных численных моделей, технология активных знаний.

Victor Emmanuilovich Malyshkin graduated from Tomsk State University with the master of sciences degree in 1970, defended his candidate dissertation in physical and mathematical sciences in Computing Centre of SB RAS in 1984, and defended his doctoral dissertation in technical sciences in Novosibirsk State University (1993). Currently is head of parallel programs synthesis laboratory in the Institute of computational mathematics and mathematical geophysics SB RAS.

Is also a founder and head of the chair of parallel computing in Novosibirsk State University. Is one of organizers of the PaCT (Parallel Computing Technologies) international conference, being held each odd year in Russia. Has more than 110 publications on parallel and distributed computing, parallel programs synthesis, supercomputing software and applications, parallel implementation of large-scale numerical models. Current scientific interests include parallel computing technologies, languages and systems of parallel computing, methods and tools of software implementation of large-scale numerical models, the active knowledge technology.

Перепелкин Владислав Александрович — научный сотрудник Института вычислительной математики и математической геофизики СО РАН; старший преподаватель каф. параллельных вычислений факультета информационных технологий Новосибирского национального исследовательского государственного университета. Тел.: (383) 330-89-94, e-mail: perepelkin@ssd.sscc.ru.

В 2008 году окончил Новосибирский государственный университет с присуждением степени магистра техники и технологии по направлению „Информатика и вычислительная техника“.

Имеет более 20 опубликованных статей по теме автоматизации конструирования параллельных программ в области численного моделирования. Является одним из основных разработчиков экспериментальной системы автоматизации конструирования численных параллельных программ для мультикомпьютеров LuNA (от Language for Numerical Algorithms).

Область профессиональных интересов включает автоматизацию конструирования параллельных программ, языки и системы параллельного программирования, высокопроизводительные вычисления.

Perepelkin Vladislav Aleksandrovich — Graduated from Novosibirsk State University in 2008 with the master degree in engineering and technology in computer science. Nowadays has the research position at the Institute of Computational Mathematics and Mathematical Geophysics (Siberian Branch of Russian Academy of Sciences), and also has a part time senior professor position in the Novosibirsk State University.

He is an author of more than 20 papers on automation of numerical parallel programs construction. He is one of main developers of fragmented programming system LuNA (Language for Numerical Algorithms).

Professional interests include automation of numerical parallel programs construction, languages and systems of parallel programming, high performance computing.

Дата поступления — 07.10.2019